TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika

Studijní obor: 1802R022 – Informatika a logistika

Návrh řídícího systému pro ovládání mobilních robotů

A design of control system for mobile robots

Bakalářská práce

Autor: **David Svoboda**

Vedoucí práce: Ing. Miroslav Holada, Ph.D.

V Liberci 16. 5. 2013

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**David Svoboda**
Osobní číslo:	**M10000046**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Informatika a logistika**
Název tématu:	**Návrh řídícího systému pro ovládání mobilních robotů**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s problematikou projektu průzkumná miniponorka na pracovišti školitele.
2. Seznamte se s možnostmi řízení, ovládání a diagnostiky mobilních robotů pomocí řídících mikrokontrolérů.
3. Navrhněte řídící systém pro nízkoúrovňové řízení a ovládání miniponorky včetně komunikace s nadřazeným počítačem.
4. Systém realizujte a prakticky otestujte.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **books.i-techonline.com - současný stav robotiky**

[2] **robotika.cz - současný stav robotiky (česky)**

[3] **Interní studijní materiály školitele**

[4] **HLAVÁČ V., SEDLÁČEK M.: Zpracování signálu a obrazu, Skripta FEL ČVUT, Praha 2000, ISBN 80-01-02114-9**

Vedoucí bakalářské práce: **Ing. Miroslav Holada, Ph.D.**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **1. října 2012**

Termín odevzdání bakalářské práce: **20. května 2013**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 - L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
pověřen vedením ústavu

V Liberci dne 1. října 2012

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce.

Datum: 16. 5. 2013

Podpis:

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu práce, panu Ing. Miroslavu Holadovi, Ph.D. za vedení práce, jeho přínosné připomínky a nápady. Dále bych chtěl poděkovat Jakubu Štěpánkovi za zapůjčení obou testovacích desek s mikrokontroléry PICAXE. Poděkování patří také všem, kteří mi poradili, když jsem potřeboval pomoci při návrhu softwaru.

Abstrakt

Cílem této bakalářské práce bylo seznámit se s řízením, ovládáním a diagnostikou mobilních robotů, s programováním mikrokontrolérů PICAXE, se způsobem návrhu komunikačního protokolu a s možnostmi realizace komunikace přes sériovou linku při využití komponent .NET, ale především pomocí funkcí Windows API za použití jazyka C++. Na základě těchto poznatků bylo hlavním úkolem navrhnout zcela nový nízkoúrovňový řídící systém pro projekt průzkumné miniponorky, který vyřeší nedostatky staršího používaného softwaru. Tento systém zahrnuje software pro mikrokontrolér i nadřazený počítač za využití navrhnutého komunikačního protokolu.

Při vypracování práce byl kladen velký důraz na rychlost a přehlednost celého systému, aby byla miniponorka schopná rychle reagovat nejen na povely obsluhy. Zároveň však byl požadavek na univerzálnost použitého komunikačního protokolu i realizované komunikace, aby se dal software využít i u jiných podobných projektů.

Komunikace po sériové lince ze strany nadřazeného počítače, napsaná v jazyce C++, byla vytvořena za použití funkcí Windows API, aby byla zajištěna dlouhodobá kompatibilita i na novějších verzích operačního systému Windows. To vyžadovalo hlubší studium této problematiky a dlouhodobé testování.

Klíčová slova: miniponorka, mikrokontrolér, PICAXE 18M2, sériový port, sériová komunikace, Windows API

Abstract

The aim of this thesis was to get acquainted with the management, control and diagnostics of mobile robots, programming microcontrollers PICAXE, the method of design communication protocol and the possibilities of realizing communication over a serial line, using components .NET, but especially by Windows API functions using the language C++. Based on these findings, the main task was to design a completely new low-level control system for project of exploration submersibles that will resolve deficiencies in older used software. The system includes software for microcontroller and superior computer using the proposed communication protocol.

During the development work has been put great emphasis on speed and clarity of the system to get the submersibles be able to quickly react to operator commands. Simultaneously there was the requirement for universality of the communication protocol and implemented communications software that could be used in other similar projects.

Communication via serial line from the superior computer written in C++ was created using the Windows API functions to ensure long-term compatibility on newer versions of the Windows operating system. This required a deeper study of this issue and long-term testing.

Keywords: minisubmarine, microcontroller, PICAXE 18M2, serial port, serial communication, Windows API

Obsah

Seznam obrázků 9

Seznam příloh 9

Seznam použitých zkratek 9

Úvod 10

1 Projekt miniponorka 11

1.1 Historický přehled 12
1.2 Řídící hardware miniponorky 14
1.2.1 Řídící (nadřazený) počítač 14
1.2.2 Mikrokontrolér Picaxe 15

2 Možnosti řízení, ovládání a diagnostiky mobilních robotů 17

2.1 Vybrané prvky řízení 17
2.2 Řízení pomocí mikrokontrolérů 18

3 Návrh řídícího softwaru 20

3.1 Komunikace přes sériový port, její vlastnosti a omezení 20
3.2 Komunikační protokol 21
3.2.1 Východisko návrhu protokolu – protokol TCP 21
3.2.2 Sestavení požadovaného komunikačního protokolu 22

4 Software pro mikrokontrolér PICAXE 18M2 25

4.1.1 Inicializační část 25
4.1.2 Zpracování příchozích dat 27
4.1.3 Odpovědi pro nadřazený počítač 29

5 COM komunikace ze strany nadřazeného počítače 30

5.1 Testovací software s komponentou SerialPort 31
5.2 Windows API 33
5.3 Asynchronní komunikace pomocí C++ a Windows API 33
5.3.1 Navázání spojení a nastavení portu 34
5.3.2 Zápis dat 35
5.3.3 Čtení dat 36

5.3.4 Ukončení spojení 37

6 Praktické otestování softwaru 38

7 Další možnosti rozvoje projektu miniponorky 39

8 Závěr 40

Bibliografie 41

Příloha A – tabulka příkazových hodnot 42

Seznam obrázků

Obr. 1 - miniponorka 11

Obr. 2 - notebooková deska v miniponorce 15

Obr. 3 - testovací deska Picaxe 18M2 16

Obr. 4 - testovací deska Picaxe 20M 16

Obr. 5 - Experimentální USB deska K8055 18

Obr. 6 - Formát segmentu protokolu TCP (4 str. 380) 22

Obr. 7 - Formát vytvořeného datového protokolu 23

Obr. 8 - Testovací software využívající komponentu SerialPort 32

Obr. 9 - ponorka při jednom z ponorů v akademickém roce 2011/2012 39

Seznam příloh

Příloha A – tabulka příkazových hodnot

Příloha B – Přiložené CD

Seznam použitých zkratek

API Application Programming Interface – rozhraní pro programování

Bd Baud – jednotka modulační rychlosti

MHz megahertz – jednotka frekvence

.NET soubor technologií v softwarových produktech

PDU protokolové datové jednotky

RAM Random-access memory – paměť s přímým přístupem

ROM Read-Only Memory – paměť pouze pro čtení

TCP Transmission Control Protocol – internetový protokol

Úvod

Tématem této bakalářské práce je seznámení se s projektem průzkumné miniponorky na pracovišti školitele a především pak realizace nového řídícího systému robota.

Celý projekt byl v průběhu minulých akademických roků zpracováván a rozšiřován ve více bakalářských projektech, pracích a diplomových projektech. V posledních třech letech se však miniponorkou zabývá poměrně stabilní tým lidí, kteří ji zdokonalují ve svých dalších akademických pracích.

Řídící systém by měl být schopný ovládat plavidlo bez komplikací a v co nejkratším čase, aby bylo možné využít plný potenciál miniponorky. Jedná se o nízkoúrovňové řízení přímo pro koncové periferie plavidla včetně komunikace pomocí vytvořeného komunikačního protokolu přes sériovou linku s nadřazeným počítačem.

Software pro mikrokontrolér PICAXE 18M2 byl realizován ve vývojovém prostředí PICAXE Programming Editor za použití programovacího jazyka BASIC.

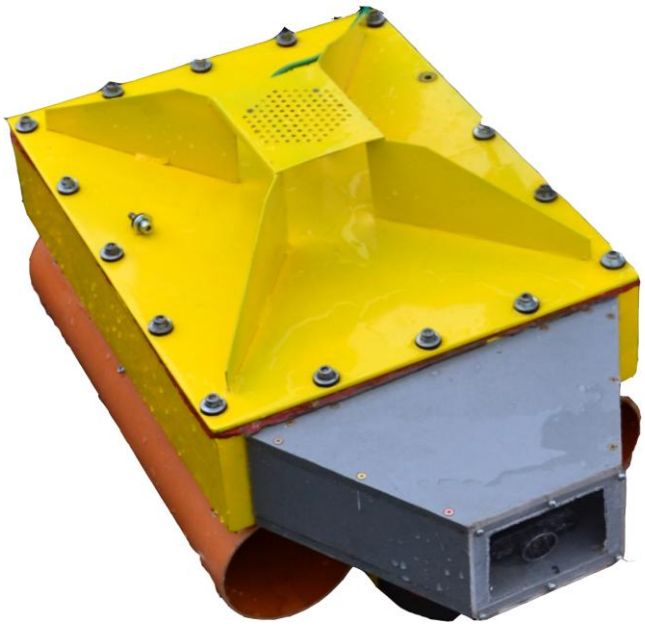
Komunikace s nadřazeným počítačem byla naprogramována v prostředí Microsoft Visual Studio 2008 v jazyce C++. Byly využity funkce Windows API, čímž se celý software obešel bez použití hotových komponent, které jsou závislé na .NET a je tak zajištěna dlouhodobá kompatibilita.

1 Projekt miniponorka

Průzkumná miniporka je plavidlo, které je schopné plout pod vodou a bylo navrženo k prozkoumávání řady vodních ploch. Jedná se o robotické zařízení ovládané obsluhou ze břehu.

Vodotěsné tělo miniponorky je rozděleno na dvě základní části:

- hlavní – zde se nachází veškerá řídící elektronika, jako jsou můstky, mikrokontroléry Picaxe, notebook a další, nalezneme zde také dvě baterie určené pro napájení, dvě balastní komory a v neposlední řadě čerpadla
- přední – tato menší část je vyhrazena pro HD kameru, která je přes USB port propojena s notebookem v hlavní části



Obr. 1 - miniponorka

Pod hlavní částí je na každé straně umístěn jeden motor. Každý z motorů je uvnitř roury, která je stejně dlouhá jako hlavní část. Tyto roury zvyšují účinnost motorů a také je chrání před poškozením. Při ponorech v rybnících, na přehradách či v jiných nečistých vodních prostorách by bez ochrany mohlo, vzhledem k malé velikosti lopatek motorů, snadno dojít k jejich poškození, a to například namotáním různých předmětů či k odlomení jejich částí při nárazu o dno, kameny, kmeny stromů apod. Roury však nemají žádná dodatečná opatření proti vniknutí

drobných nečistot, mezi které při zkušebních ponorech v minulých letech patřilo zejména listí a větvičky. Po nějaké době docházelo k jejich zanesení, čímž nastala nutnost je vyčistit, protože tím ponorka citelně ztrácela na manévrovací schopnosti.

Na spodním okraji přední části je instalováno šest LED diod pro osvětlení. Mnohem více LED diod najdeme ještě níže a jsou určeny jako hlavní světlo. Bohužel i při použití velkého množství světelných zdrojů není intenzita světla dostačující a nepřináší tedy příliš velké zlepšení obrazu, který snímá kamera. Také zapojení a umístění není v současnosti dobře vyřešeno, neboť došlo k poruchám na horním osvětlení a díky špatnému těsnění u spodního osvětlení nastal zkrat, který LED diody zcela vyřadil z provozu. Při následujícím vývoji projektu se bude muset vybrat vhodnější zdroj světla, u kterého bude potřeba zohlednit jak svítivost, tak také spotřebu energie vzhledem k tomu, že celá miniponorka je napájena z baterií. Samozřejmostí je pak přepracování umístění a zajištění dostatečné vodotěsnosti.

1.1 Historický přehled

Projekt miniponorky vznikl na základě myšlenky vedoucího práce pana Ing. Miroslava Holady, Ph.D. v roce 2009. V průběhu let se na projektu vystřídalo několik studentů, kteří vytvořili první návrhy, 3D model ponorky, samotnou konstrukci a prvotní ovládací software, který byl napsán v Delphi. Postupně byla ponorka vylepšována - například výměna kamery, H-můstků a dalších elektrických zařízení, ale i obměna softwaru. I do budoucna se počítá s dalším vývojem.

Následující seznam prací mapuje vývoj ponorky v uplynulých letech:

Akademický rok 2009/2010:

Jiroušek Petr: Průzkumná miniponorka - návrh a realizace ovládacího softwaru

Peklák Martin: Průzkumná miniponorka - návrh a realizace systému napájení

Oliva Jan: Průzkumná miniponorka - návrh a realizace kamerového systému

Roubíček Miroslav: Průzkumná miniponorka - návrh a realizace elektropohonu

Akademický rok 2010/2011:

Peklák Martin: Návrh elektronické výbavy experimentální dálkově řízené miniponorky

Roubíček Miroslav: Návrh konstrukce experimentální dálkově řízené průzkumné miniponorky

Vričan Ondřej: 3D Vizualizace průzkumné miniponorky

Akademický rok 2011/2012:

Peklák Martin: Projekt miniponorka - možnosti statického a dynamického dovažování

Svoboda David: Nízkoúrovňový řídící software

Štěpánek Jakub: Elektronická výbava mobilního robota

Zajíc Tomáš: Grafický ovládací terminál

Akademický rok 2012/2013:

Peklák Martin: Systém vyvažování průzkumné miniponorky

Svoboda David: Návrh řídícího systému pro ovládání mobilních robotů

Štěpánek Jakub: Bezdrátový komunikační modul pro mobilní roboty

Zajíc Tomáš: Návrh interaktivního softwaru pro vzdálené ovládání mobilních robotů

I pro další akademické roky počítá pan Ing. Miroslav Holada, Ph.D. s rozvojem projektu. Mělo by se jednat například o bezdrátové řízení, celkovou úpravu ponorky, sjednocení softwaru nebo přidání různých měřících čidel.

1.2 Řídící hardware miniponorky

Každý složitější robot je zpravidla ovládán počítačem. Může se jednat o větší zařízení - klasické PC nebo notebook. Ty však nelze v mnoha případech použít kvůli nedostatku prostoru, možnosti chlazení, napájení atd. V takovém případě lze využít tzv. mikrokontroléry. Mikrokontroléry obsahují mikroprocesor s vnitřní pamětí a obvody rozhraní. Stejně jako PC zpracovávají soubor instrukcí a data mají uložena v pamětích typu FLASH, čímž se výrazně šetří prostor. Na druhou stranu mikrokontroléry jsou například oproti PC obtížně rozšiřitelné, což může vést k použití více mikrokontrolérů v jednom robotovi, ale naštěstí to není finančně příliš nákladné. V miniponorce jsou použity dva druhy počítačů – notebook a mikrokontrolér.

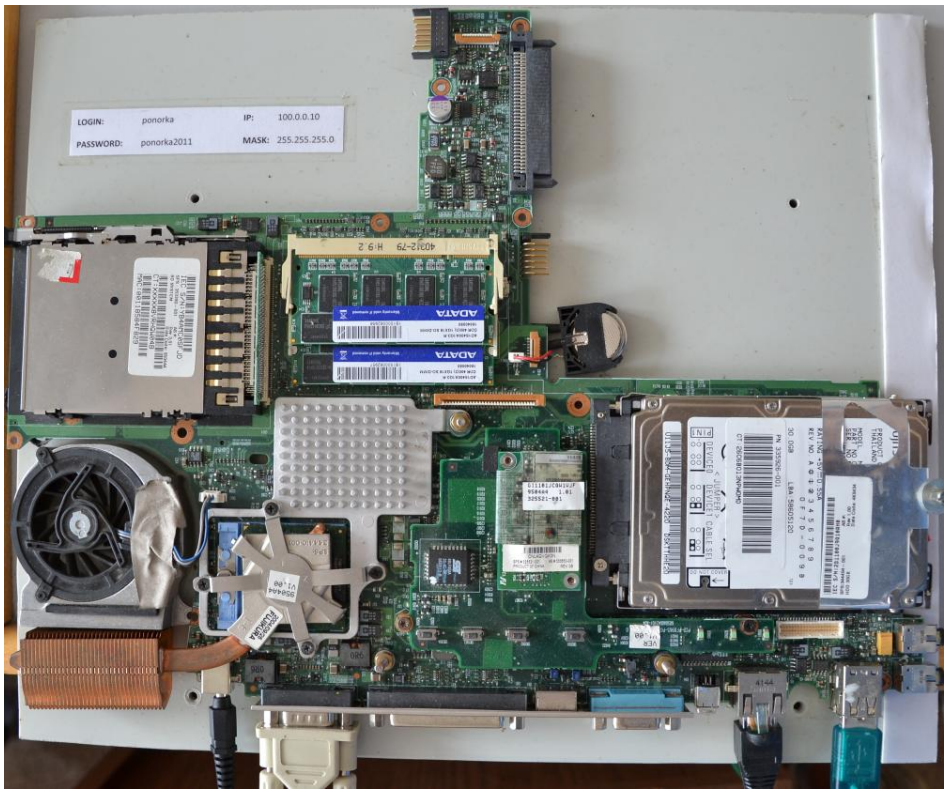
1.2.1 Řídící (nadřazený) počítač

V ponorce není samozřejmě umístěn celý přenosný počítač, ale pouze jeho část, a to konkrétně základní deska se všemi potřebnými komponentami – pevný disk s kapacitou o velikosti 30GB, dva moduly pamětí RAM, přičemž každý z nich disponuje pamětí 1GB (celkem tedy má notebook k dispozici 2GB RAM) a procesor Intel Pentium M s taktovací frekvencí 1,4 GHz. Najdeme zde také chlazení, čtyři ovládací tlačítka a několik konektorů – 2x USB, VGA, LAN, sériový port, napájení a další.

Celá tato deska pochází pravděpodobně z notebooku HP Compaq nx5000 Business. Jedinou nepůvodní komponentou jsou paměti RAM. Do budoucna se však plánuje kompletní výměna notebooku za výkonnější. Nainstalovaným operačním systémem jsou Windows XP Professional v 32-bitové verzi.

Notebook je v pozici nadřazeného počítače nad mikrokontrolérem a zprostředkovává spojení mezi ponorkou a obsluhou na břehu. K ovládání ponorky je v současnosti využíváno více softwarů (pro povely, video a údaje o náklonu miniponorky a stavu naplnění balastních komor), z nichž některé jsou spouštěny přímo na tomto notebooku a je potřeba je ovládat pomocí vzdálené plochy. Ostatní

jsou rozděleny na dvě části – server a klient, přičemž uvnitř ponorky je spuštěn server a ze břehu se k němu připojuje obsluha pomocí klientské aplikace.

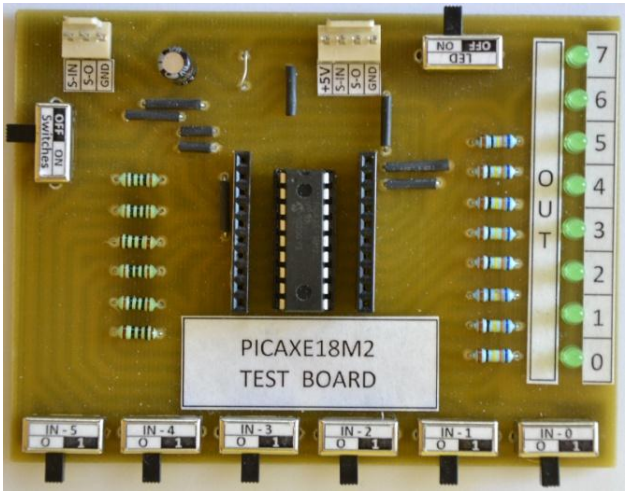


Obr. 2 - notebooková deska v miniponorce

1.2.2 Mikrokontrolér Picaxe

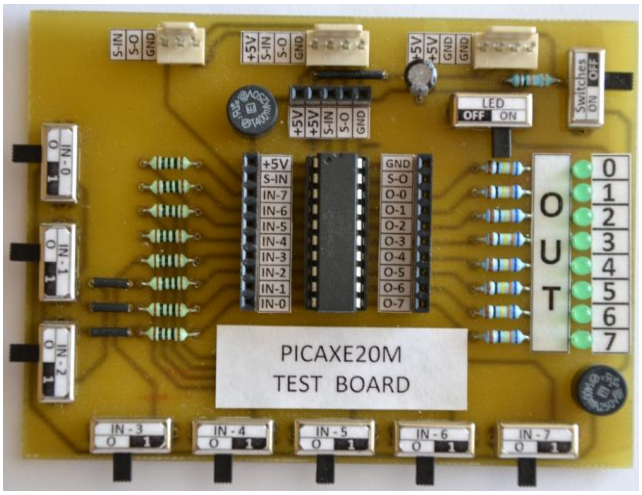
Mikrokontroléry Picaxe byly původně navrženy jako výukové systémy. Postupně se však rozšířily mezi amatérské elektroniky, modeláře a robotiky, ale také i mezi profesionály. Jejich hlavní výhodou je nízká pořizovací cena. Mají zdarma přístupnou dokumentaci a vývojové prostředí, které používá jednoduchý, snadno pochopitelný a i přesto pro tyto účely výkonný programovací jazyk BASIC. Navíc je k dispozici celá řada typů Picaxe, které se liší především velikostí paměti, počtem vývodů (většina z nich může být libovolně konfigurována na vstup či výstup) a pracovní frekvencí.

V ponorce je použit typ Picaxe 18M2, který podporuje až 16 vstupů/výstupů s 10 analogovými kanály. Pracovní frekvence je ve výchozím stavu 4 Mhz, dokáže však pracovat na frekvenci až 32 Mhz. V úvodní fázi vývoje nového softwaru byl však použit i Picaxe 20M, který je oproti 18M2 značně omezený, ale pro testovací účely postačoval (více viz kapitola 4).



Obr. 3 - testovací deska Picaxe 18M2

Množství ovládaných prvků v miniponorce však již během posledního vývoje projektu přesáhlo možnosti jednoho mikrokontroléru, a tak bylo nutné při návrhu protokolu a softwaru zohlednit i to, že jich bude použito více. V současnosti se pracuje se dvěma, ale ani to zřejmě nebude brzy stačit. Jako dalšími možnostmi vývoje se nabízí přidání dalších mikrokontrolérů, které budou i s těmi stávajícími různě adresovány a nebudou tedy vykonávat „cizí“ příkazy, nebo výměna za větší typ mikrokontroléru (například Picaxe řady 40), který bude zvládat ovládat všechny periferie ponorky.



Obr. 4 - testovací deska Picaxe 20M

2 Možnosti řízení, ovládání a diagnostiky mobilních robotů

Mobilní roboti se rozdělují podle několika různých kritérií, především však na dva základní typy – autonomní a dálkově řízené. U prvně jmenovaných se předpokládá schopnost samostatně vykonávat zadanou činnost. Může se jednat například o sledování barevné čáry na podložce a schopnost reagování na případné překážky, nebo se umět pohybovat v neznámém prostředí – tedy zmapovat ho, orientovat se v něm a poté dosáhnout stanoveného cíle. Oproti tomu dálkově řízení roboti jsou ovládáni operátorem, který má zpravidla vizuální ovládací software a zejména vizuální informace o pracovním okolí robota. Přesto i takto řízený robot by měl být částečně autonomní, aby například při ztrátě spojení byl schopen se vrátit na výchozí bod či na jiné, pro obsluhu dostupné místo. (1 str. 11)

2.1 Vybrané prvky řízení

Roboti se v současnosti řídí především:

- klasickými PC
- notebooky
- mikrokontroléry
- dalšími zařízeními – PDA, mobilní telefony, …

Často se v řízení robotů využívá kombinace některých z výše uvedených zařízení. Důkazem toho je i miniponorka, která kombinuje notebook a mikrokontrolér.

Klasické PC se u robotů většinou využívá všude tam, kde to dovolují prostory a také tam, kde je potřeba postupného upgradu komponent. Jejich hlavními výhodami pak jsou vysoký výkon, snadná dostupnost jednotlivých dílů anebo administrace přes známá rozhraní, jako je například operační systém Windows či Linux. Na druhou stranu mají zpravidla velké rozměry a vyšší spotřebu elektrické energie a je také potřeba dostatečné chlazení jednotlivých komponent.

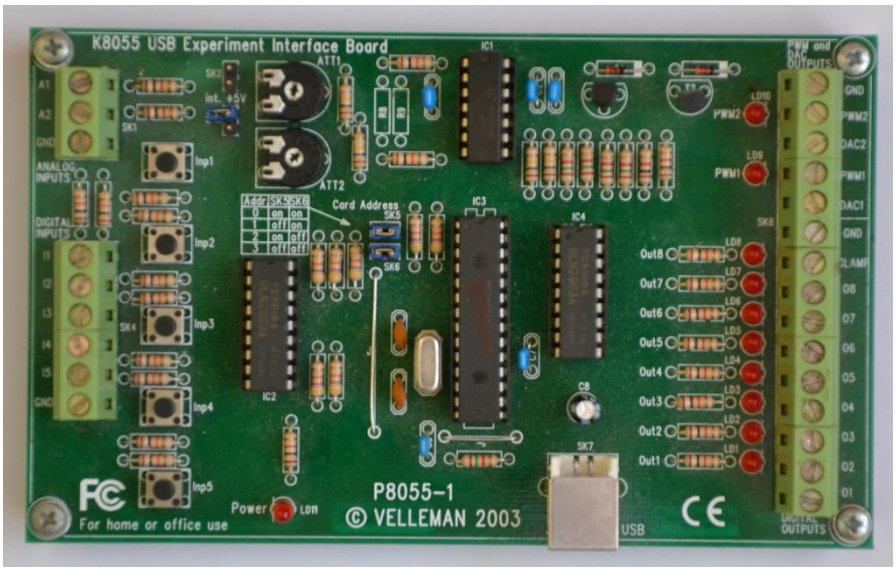
Notebooky jsou prakticky zmenšená PC, které v robotech můžeme opět využít například tam, kde chceme používat operační systém MS Windows, Linux a podobně. Zároveň však požadujeme nižší spotřebu a menší velikost oproti PC. Proto byl notebook vybrán do ponorky, což umožnilo především vytváření aplikací, které jsou určeny pro operační systém Windows.

Pokud je potřeba vytvořit malého robota, bude pravděpodobně nejlepší jako centrální mozek použít mikrokontroléry.

2.2 Řízení pomocí mikrokontrolérů

Mikrokontroléry představují v podstatě velmi jednoduchý, ale kompletní mikropočítačový systém. Jejich vnitřní struktura obsahuje všechny potřebné bloky. Mají generátor hodin, paměť ROM i RAM, obvody vstupů a výstupů, čítače, časovače a sériový kanál. Rozsah pamětí je sice malý, ale dostatečný pro úkoly, pro které jsou obvody používány. Vyznačují se nízkou cenou, vysokou spolehlivostí, malými rozměry a malou energetickou náročností.

Existuje mnoho různých mikrokontrolérů. Mezi nejznámější a nejpoužívanější se řadí mikrokontroléry Atmel, Picaxe, PIC a 8051 či 8055. V miniponorce byla v minulosti použita experimentální USB deska 8055, v současné době se již využívají mikrokontroléry Picaxe typu 18M2.



Obr. 5 - Experimentální USB deska K8055

Mikrokontroléry nejčastěji využívají tzv. TTL logiku. Ta definuje, že napětí 5V reprezentuje logickou jedničku a napětí 0V naopak reprezentuje logickou nulu. Stačí tedy, když se vstupní piny chovají jako jednoduché voltmetry, výstupní jako zdroje napětí a čip může komunikovat s okolím.

Jednoduché vstupní piny nám umožní pouze detekovat stav zapnuto/vypnuto. Stejně tak jednoduché výstupní piny nám umožní vyslat pouze 0V nebo 5V. Jsou ale situace, kdy je potřeba přesně vědět, jaké napětí je přivedeno na vstupní pin. K tomu slouží analogové vstupy. Tyto vstupy mají A/D převodník, který je schopen přesněji změřit příslušné napětí. Toto napětí je převedeno na číslo, které je k dispozici programu (typicky je to hodnota v rozsahu 0 - 255, která přímo úměrně odpovídá vstupnímu napětí 0 - 5V). Stejně tak mohou být D/A výstupy, které dokážou na výstupním pinu generovat různá napětí.

Další proměnnou, která hraje klíčovou roli, je čas. Není důležitá pouze aktuální hodnota napětí na jednotlivých pinech, ale i jak se mění v čase. Čas je opět diskrétní. Při frekvenci 10MHz robot svět „pozoruje“ každých 1/10 000 000 sekund, tedy 100ns. (2)

3 Návrh řídícího softwaru

Stěžejní částí této bakalářské práce bylo navrhnout samotný řídící software průzkumné miniponorky. Tento SW je rozdělen na dvě části:

- komunikace nadřazeného počítače přes sériový port typu RS232 s řídícím mikrokontrolérem (připojení, čtení a zápis dat)
- program pro mikrokontrolér Picaxe 18M2

Větší a složitější částí celé práce byla komunikace přes sériový port mezi dvěma zařízeními pod Windows a podrobně se jí věnuje kapitola 5.

Celková koncepce řídícího softwaru vycházela jednak z podnětů vedoucího práce pana Ing. Miroslava Holady, Ph.D., ale také z poznatků získaných během zpracovávání bakalářského projektu v akademickém roce 2011/2012. Hlavní pozornost tedy byla věnována především komunikaci mezi notebookem, resp. serverovou aplikací, a Picaxem po sériové lince. Zde bylo totiž oproti předchozím verzím ovládacího softwaru nutné zavést režim „příkaz – odpověď“, aby nedocházelo například k tomu, že pokud se některý příkaz špatně zpracuje, vůbec se neprovede nebo se při přenosu ztratí, tak obsluze není nijak oznámena chyba a ani není proveden opětovný pokus o úspěšné vykonání požadovaného příkazu.

3.1 Komunikace přes sériový port, její vlastnosti a omezení

Při komunikaci přes sériový port se data přenáší po jednom bitu. Sériový port je odolný proti rušení a přenos může probíhat až na vzdálenost 15 metrů při rychlosti 19 200 Bd, přičemž se snižující se rychlostí může být maximální přenosová vzdálenost větší. (3 str. 16) V projektu miniponorky se ale tato vzdálenost řešit nemusí, jelikož notebook se sériovým portem typu RS232 je umístěn přímo uvnitř těla plavidla a délka spojovacího kabelu je tak zkrácena na minimum – několik centimetrů.

Sériový port PC používá asynchronní přenos. Ten začíná tzv. start-bitem, následují datové bity, dále může být paritní bit a rámec je ukončen stop-bitem. Přijímač a vysílač musí být nastaveny na stejnou přenosovou rychlost a synchronizace obou obvodů proběhne pomocí start-bitu. Stop-bit pak vytváří

prodlevu nutnou pro zpracování dat přijímačem a vyhodnocení úspěšnosti přenosu. (3 str. 16)

Pro komunikaci počítače přes sériovou linku je nutné nastavit správné parametry přenosu. Jsou to:

- rychlost udávaná v Bd (Baud)
- počet datových bitů (5 až 8)
- parita (sudá, lichá, žádná)
- délka stop-bitu (1; 1,5 nebo 2)

3.2 Komunikační protokol

Komunikační protokol je jádrem celé komunikace. Je obecně definován jako množina pravidel syntaktických (formáty jednotlivých zpráv, tj. příkazů a odpovědí) a sémantických (používání příkazů a odpovědí), která určují chování funkčních jednotek při nějaké činnosti včetně časové specifikace výskytu jednotlivých událostí nebo jejich posloupností. Specifikace protokolu musí být pro zajištění kompatibility přesně definovaná. Protokol je definován protokolovými datovými jednotkami a jejich formátem – například délkou, kódem, abecedou nebo uspořádáním, pravidly pro výměnu protokolových datových jednotek (PDU) mezi entitami a hodnotami měřitelných a nastavitelných veličin, parametrů, jako je např. časová prodleva, počet opakování, objem uživatelských dat, šíře okna, modul číslování apod., které lze dohodnout mezi komunikujícími entitami před vlastní komunikací. (4 str. 47)

V zadání této bakalářské práce bylo navrhnout komunikaci nadřazeného počítače s mikrokontrolérem a k tomu tedy bylo potřeba vytvořit i komunikační protokol. Celkový návrh vychází z podnětů vedoucího práce.

3.2.1 Východisko návrhu protokolu – protokol TCP

Při návrhu se nabízela možnost inspirovat se nějakým již vytvořeným, v praxi zavedeným a rozšířeným protokolem. Toto východisko bylo ve spolupráci s vedoucím práce nalezeno v Transmission Control Protocol, tedy v tzv. TCP. Jeho architektura se stala jednou z nejvyužívanějších v oblasti síťové komunikace.

TCP je transportní protokol se spojením. Poskytuje logické spojení mezi koncovými aplikacemi, tedy spolehlivý přenos dat, který nebyl zajištěn diagramově orientovaným protokolem IP. TCP je bytově-orientovaný protokol.

Jedním z hlavních principů TCP je nezávislost na nižší infrastruktuře, protože na ni spoléhá jen minimálně. Aplikační vrstvě poskytuje TCP službu plného duplexu, tj. data se mohou posílat v obou směrech nezávisle na sobě. (4 str. 380)

<table>
<tr><td colspan="2">zdrojový port</td><td colspan="2">cílový port</td></tr>
<tr><td colspan="4">pořadové číslo</td></tr>
<tr><td colspan="4">číslo potvrzení</td></tr>
<tr><td>délka záhlaví =
násobek 32 bitů</td><td>rezervováno
= 0</td><td>funkce
řízení</td><td>šířka okna</td></tr>
<tr><td colspan="2">kontrolní součet</td><td colspan="2">označení urgentních dat</td></tr>
<tr><td colspan="4">volitelné možnosti (příp. doplnění do násobků 32 bitů)</td></tr>
<tr><td colspan="4">data</td></tr>
</table>

Obr. 6 - Formát segmentu protokolu TCP (4 str. 380)

3.2.2 Sestavení požadovaného komunikačního protokolu

Při navrhování komunikace po sériové lince, a tedy komunikačního protokolu, byly jasně stanoveny dva základní požadavky:

- režim „příkaz – odpověď“
- smysluplnost a srozumitelnost

Režim „příkaz – odpověď“ spočívá v této práci v tom, že nadřazený počítač vyšle řídícímu mikrokontroléru paket dat s instrukcemi. Pokud data dorazí, vyšle se zpět datový paket s odpovědí, čímž se získá informace o tom, jak a zda vůbec byla data zpracována. Tyto odpovědi jsou popsány v kapitole 4.1.3.

Smysluplností a srozumitelností se nemá na mysli pouze pochopení protokolu ze strany počítačové, ale také ze strany obsluhy, resp. programátora. Komunikační protokol byl navrhován pro univerzální použití, tedy nejen v miniponorce, ale i v jiných robotech, kteří komunikují po sériové lince.

Jak bylo popsáno v předchozí podkapitole, protokol realizovaný v této bakalářské práci byl inspirován zavedeným protokolem TCP. Celá koncepce však byla značně zjednodušena, jelikož tolik komplexní protokol, jako je TCP, je záležitost dlouhodobého vývoje a navíc obsahuje spoustu nadbytečných prvků pro potřeby komunikace uvnitř miniponorky mezi nadřazeným počítačem a mikrokontrolérem. Jak ilustruje následující obrázek, formát je podstatně zmenšený oproti TCP.

délka zprávy	data	ID

Obr. 7 - Formát vytvořeného datového protokolu

První část – délka zprávy – obsahuje informace o tom, kolik bytů se odesílá a má tedy i být přijato. V projektu miniponorky byla zvolena délka 12 bytů, přičemž pro každou část jsou přiděleny čtyři byty.

Datová část obsahuje zasílaná data. Každý byte v celém protokolu má maximální hodnotu 256, proto se v průběhu práce s tímto protokolem musí čísla instrukcí vyšší než limit přepočítávat a následně rozdělit do více bytů. Vzhledem k tomu, že zasílané hodnoty jsou v současném softwaru miniponorky čtyřmístné, jejich přepočet se rozdělil do dvou bytů a zbylé dva byty datové části zůstaly nulové. Přepočet čísla, například 2000, se provádí tak, že se číslo vydělí hodnotou 256 a výsledek se zaokrouhlí na celé číslo dolů, což je v tomto případě 7. Tím získáme hodnotu do druhého bytu. Hodnota do prvního bytu se pak dostane z rozdílu zadaného čísla a násobku hodnoty 256, spočítané v předchozím kroku. V tomto případě se tedy obsah prvního bytu dat vypočte jako 2000 – 7*256 = 208.

Poslední čtyři byty protokolu jsou vyhrazeny pro identifikaci zprávy, která je vyjádřena číselným identifikátorem (ID). To slouží například k tomu, aby se dalo snadno určit, na kterou zprávu se vrátila odpověď. Bez této informace by mohlo dojít například k situaci, kdy obsluha vyšle příkaz k zapnutí motorů a krátce na to příkaz k jejich zastavení. Vlivem okolností by však k zastavení motorů nedošlo (ztráta dat při přenosu, změna dat vlivem šumu, atd.), a i přesto by dorazila odpověď o úspěšném vykonání operace, která by ovšem příslušela první instrukci o zapnutí motorů. Identifikátor však může v sobě nést i informaci o tom, kterému zařízení je celá zpráva určena, což může být užitečné, pokud se ovládá více

připojených zařízení najednou. V konkrétním případě miniponorky se bude jednat o několik mikrokontrolérů.

Jak bylo napsáno výše, v projektu miniponorky se přistoupilo k 12-ti bytové délce zprávy. Jedna ze sestavených zpráv, odeslaná dle formátu použitého protokolu, pak vypadá takto:

délka zprávy				data				ID			
12	0	0	0	209	7	0	0	1	0	0	0

V tomto případě se jedná konkrétně o instrukci pro zapnutí motorů ve směru vpřed. Číselná hodnota této instrukce je 2001, z čehož vyplývá, že musela být přepočítána a rozdělena do dvou datových bytů obsahujících hodnoty 209 a 7.

Číselné hodnoty příkazů zasílaných miniponorce navrhl v akademickém roce 2012/2013 ve své bakalářské práci Tomáš Zajíc.

4 Software pro mikrokontrolér PICAXE 18M2

Na základě navrženého protokolu (viz kapitola 3.2.2) bylo potřeba zhotovit zcela nový software pro mikrokontrolér. Ten již pracuje na bázi „příkaz – odpověď“ na rozdíl od předchozích verzí softwaru, který se používal v minulých letech v projektu miniponorky. Starší software pouze přijal data, a pokud byla správná, instrukci vykonal. Obsluha neměla jinou než vizuální kontrolu o tom, zda se provedla požadovaná činnost.

Software pro mikrokontrolér je napsán v jazyce BASIC. V tomto jazyce lze psát kód přímo ve vývojovém editoru PICAXE Programming Editor. Editor obsahuje i mód simulace, kdy není potřeba mít připojen mikrokontrolér a i tak je možné ladit chyby či případné nedostatky rychle a efektivně.

Před začátkem vývoje celého softwaru pro Picaxe 18M2 bylo potřeba seznámit se problematikou programování mikrokontrolérů a také s programovacím jazykem BASIC. Tyto kroky byly provedeny na zapůjčené testovací desce osazené typem PICAXE 20M, který se od v ponorce použitého typu 18M2 liší především tím, že disponuje pamětí pouze 256 bytů (cca 80 řádků programu) oproti 2048 bytům (cca 1800 řádků programu). Taktovací frekvence je maximálně 8MHz oproti 32MHz a je zde také úplná absence vstupně / výstupních bitů. Pro testovací účely a jednoduché programy vytvořené za účelem porozumění, především syntaxe jazyka BASIC, však i tento starší model 20M postačoval.

4.1.1 Inicializační část

V inicializační části je potřeba přiřadit jednotlivé připojené součásti miniponorky na vstupy a výstupy mikrokontroléru. To zajišťuje následující kód:

```
symbol MRR      =    B.7 '(13)    - Motor_Right_Reverse
symbol MRF      =    B.6 '(12)    - Motor_Right_Forward
symbol PRF      =    B.5 '(11)    - Pump_Right_Forward
symbol PRR      =    B.4 '(10)    - Pump_Right_Reverse
symbol MLF      =    B.3 '( 9)    - Motor_Left_Forward
symbol MLR      =    B.2 '( 8)    - Motor_Left_Reverse
```

```
symbol LIGHT_1 =    C.7 '(16)    - OUT_Lights1

symbol LIGHT_2 =    C.6 '(15)    - OUT_Lights2

symbol MSG_INPUT_PORT = C.5 ' - com_serial_input

symbol MSG_OUTPUT_PORT = C.0 '- com_serial_output

symbol BATT_1  =    C.1 '(18)    - IN_ADC_battery1_voltage

symbol BATT_2  =    C.0 '(17)    - IN_ADC_battery2_voltage

symbol VOLTAGE1=    b2  'battery1 voltage

symbol VOLTAGE2=    b3  'battery2 voltage
```

Symboly jsou v inicializační části použity záměrně z toho důvodu, aby bylo možné velice rychle upravit výstupní nebo vstupní pin, pokud by například bylo potřeba změnit vnitřní zapojení součástí ponorky.

Jelikož mikrokontrolér komunikuje s nadřazeným počítačem prostřednictvím sériové linky, je nutné nastavit parametry sériového přenosu. Tyto parametry není možné změnit ovládacím softwarem ze břehu a jsou pevně stanoveny jako optimální. Nastavení v nadřazeném počítači se tedy musí přizpůsobit. Parametry jsou v mikrokontroléru nastaveny následujícím kódem v inicializační části:

```
symbol BAUDMODE= N9600_8
```

Opět je využit symbol, a to ze stejného důvodu jako v předchozích případech. Konkrétní použití toho symbolu je vidět v kódu v kapitole 4.1.2 a 4.1.3. Jak je ze zápisu symbolu patrné, přenosová rychlost byla stanovena na 9600 Bd a počet datových bitů na osm.

Ve výchozím stavu je PICAXE 18M2 nastaven na pracovní frekvenci 4 MHz. Při této frekvenci je však schopen pracovat s baudrate maximálně 4800 Bd, přičemž počet datových bitů je pak čtyři. Protože jsou ale obě hodnoty požadovány vyšší, je potřeba i vyšší pracovní frekvence mikrokontroléru, a to konkrétně 8 MHz, u které je maximální baudrate právě 9600 Bd a počet datových bitů osm. Přetaktování zajišťuje jednoduchý příkaz:

```
setfreq m8
```

Použitý mikrokontrolér PICAXE 18M2 dokáže pracovat ještě na frekvenci 16 MHz, nejvyšší pracovní frekvence je 32 MHz – maximální baudrate 38 400 Bd, počet datových bitů 32. Takové frekvence jsou však pro potřeby této komunikace zbytečné.

Celá inicializační část je zakončena odesláním `SendMSG_OK`, což je informace o úspěšném vykonání první části programu.

4.1.2 Zpracování příchozích dat

Po inicializaci přichází na řadu příjem dat. Pro čtení dat ze sériového portu se v jazyce BASIC využívá funkce `serin`, která má v realizovaném softwaru využity tři hlavní parametry – pin, baudmode a qualifer. Jako další parametr lze u té funkce využít timeout jenž určuje, po jak dlouhou dobu se má na sériovém portu naslouchat. To zde ale nemůže být využito, protože je potřeba, aby mikrokontrolér stále vyčkával na příchod dat, jelikož není předem známo, v jaký okamžik budou data zaslána.

Příjem dat je v programu označen jako `ReadMSG` a celý zápis vypadá takto:

```
serin MSG_INPUT_PORT, BAUDMODE, B20, B21, B22, B23, B24, B25, B0, B0, B26, B27, B0, B0
```

První parametr je tedy vstupní port a druhý je přenosová rychlost sériového portu, přičemž tyto dva parametry byly nastaveny v inicializační části. Následuje zápis 12-ti tzv. qualiferů. Tyto qualifery ukazují na místa v paměti mikrokontroléru, kam se přijatá data uloží. Jelikož je v současnosti dopředu známo, že sedmý, osmý, jedenáctý a dvanáctý byte přijaté zprávy bude vždy mít hodnotu nula, není každému z nich přiřazeno zvláštní místo v paměti, ale je využito pouze jedno, a to B0. Pokud jsou data úspěšně přijata, je opět odeslána zpráva `SendMSG_OK`.

Pro následující část programu jsou důležité byty uložené na adrese B24 a B25, což jsou první dva datové byty přijaté zprávy. Picaxe umožňuje přístup k hodnotám po jednotlivých bytech, ale také umí byty složit do datového typu word, tedy dva byty dohromady. Přístup k nim se provádí přes proměnné `wi`, kde první písmeno označuje datový typ word a druhé je nahrazeno pořadovým číslem.

Například bytům na adresách B0 a B1 odpovídá word W0, byty B1 a B2 se dají vyjádřit jako W1 a tak dále. K požadovaným bytům na adresách B24 a B25 se tedy v programu přistupuje přes word W12. Následující ukázka kódu ilustruje práci s hodnotou uloženou v W12 a další vykonávané instrukce:

```
if w12=2001 then; pohyb VPRED
          low MRR
          low MLR
          high MRF
          high MLF
          gosub SendMSG_DONE

elseif w12=2002 then; pohyb VZAD
          low MRF
          low MLF
          high MRR
          high MLR
          gosub SendMSG_DONE
```

První řádek je zápis podmínky, ve které porovnáváme, zda je ve wordu W12 uložená hodnota 2001, která je přidělena pohybu vpřed. Pokud ano, vykoná se pět následujících příkazů a to konkrétně:

a) vypnutí zpětného chodu pravého i levého motoru (pokud zpětný chod nebyl zapnut, nic nebrání tomu jej i tak deaktivovat, předejde se tím sepnutí motorů do obou směrů, což by mohlo vést k problémům)
b) zapnutí chodu vpřed u obou motorů
c) poslední příkaz zavolá `SendMSG_DONE`, aby nadřazený počítač mohl uživateli předat zprávu, že zaslaná instrukce byla vykonána (více viz kapitola 4.1.3)

V případě, že ve wordu W12 je uložená jiná hodnota než 2001, podmínka je vyhodnocena jako FALSE a program pokračuje na další podmínku, kde je požadována hodnota 2002 a tak dále. Tabulka všech hodnot je v příloze A. Žádná z podmínek nemusí být vyhodnocena jako TRUE (například pokud jsou přijata neúplná data nebo mohou být i odlišná). V takovém případě se odešle

`SendMSG_ERR`. Na konci celého zpracovávání a vyhodnocení přijatých dat se program vrací zpět na začátek, kde čeká v `ReadMSG` na nová data.

4.1.3 Odpovědi pro nadřazený počítač

V průběhu vykonávání programu dochází k odesílání různých zpráv a odpovědí nadřazenému počítači na základě přijatých požadavků. Tyto odpovědi mohou být aktuálně čtyři:

- OK – připraveno k další činnosti, kód zprávy 100
- DONE – instrukce byla vykonána, kód zprávy 999
- ERR – došlo k chybě, kód zprávy 500
- BATT – zasílání informací o aktuálním stavu baterií

Tvar zprávy je složen podle komunikačního protokolu z 12-ti bytů. Zprávy OK, DONE a ERR mají pevně daný tvar s proměnlivým 9. a 10. bytem (B26 a B27), které budou sloužit k identifikaci zprávy (viz kapitola 3.2.2). Zápis zprávy `SendMSG_DONE` vypadá takto:

```
serout MSG_OUTPUT_PORT, BAUDMODE, (12, 0, 0, 0, 231, 3, 0, 0, B26, B27, 0, 0)
```

Pokud dojde k zavolání `SendMSG_DONE`, je odeslána zpráva na sériový port, nastavený v inicializaci při určené přenosové rychlosti udané ve stejné části.

Od výše popsaných zpráv se pak drobně liší `SendMSG_BATT`, která se skládá z odeslání dvou datových paketů mající proměnlivý celkově pátý, resp. první datový byte (B24), ve kterém je předána hodnota `VOLTAGE1`, resp. `VOLTAGE2`, udávající hodnotu aktuálního stavu baterie 1, resp. baterie 2.

5 COM komunikace ze strany nadřazeného počítače

Komunikace po sériové lince ze strany nadřazeného počítače je oproti komunikaci ze strany mikrokontroléru značně složitější. Přesněji řečeno, programování je složitější. Zatímco v jazyce BASIC stačí použít několik příkazů a takový program je schopen fungovat i za několik let (pokud bude nahrán do stejné verze mikrokontroléru), programování pod Windows tak jednoduché není.

Požadavkem vedoucího práce bylo, aby celé komunikování pod Windows bylo řešeno pomocí jazyka C++. Preferováno bylo použití funkcí Windows API před využitím komponent .NET. Pro realizaci softwaru bylo použito vývojové prostředí Microsoft Visual Studio 2008, jenž je k dispozici zdarma v programu MSDN.

Z počátku vývoje nového ovládacího softwaru pro nadřazený počítač byla vytvořena jednoduchá aplikace Windows Forms. Ta využívá pro komunikaci přes sériovou linku předpřipravenou komponentu SerialPort, která je v tomto prostředí u vizuálního návrhu aplikace k dispozici. A právě ta na začátku vývoje usnadnila práci a přispěla k lepšímu porozumění problematiky. Tuto fázi vývoje popisuje kapitola 5.1.

V druhé fázi se začal postupně vyvíjet program s vlastními funkcemi v nativním C++. Ve výsledném programu tedy nebyly použity žádné komponenty, ale pouze funkce Windows API.

V obou případech se jedná o komunikaci v režimu se spojením, což znamená, že je rozdělena na tyto fáze:

- navázání spojení
- přenos dat
- ukončení spojení

Výhodou tohoto režimu je, že se dají předem určit parametry a volby pro daný přenos. Také se obvykle používá potvrzování přijatých dat, aby se zajistila větší pravděpodobnost detekce případných ztrát. Detekce chyb se provádí nejčastěji formou kontrolního součtu dat a je součástí i zde používaného protokolu.

5.1 Testovací software s komponentou SerialPort

V počátku vypracovávání této bakalářské práce byla hlavní pozornost věnována vývoji řídícího programu pro mikrokontrolér a komunikačnímu protokolu. Aby se dala spolupráce nadřazeného počítače s mikrokontrolérem otestovat, bylo potřeba vytvořit alespoň jednoduchý program pro komunikaci po sériové lince. Ten byl vytvořen za použití komponenty SerialPort, kterou lze ve vývojovém prostředí Microsoft Visual Studio 2008 velice snadno použít. Zároveň tento testovací software poskytl názornou ukázku toho, jak se celá komunikace chová a z těchto poznatků bylo čerpáno při vytváření finálního softwaru za využití Windows API.

Komponenta SerialPort pracuje v synchronním režimu I/O. To znamená, že systém zašle zprávu druhému zařízení a vyčkává na odpověď. Pokud jsou data přijata, ihned jsou zpracována. Cílem bakalářské práce však bylo udělat asynchronní komunikaci (viz kapitola 5.3). Pro testování však postačil synchronní přenos.

Použití komponenty je poměrně snadné. Ve vývojovém prostředí Microsoft Visual Studio je k dispozici menu, ze kterého lze přímo nastavit veškeré potřebné parametry sériového přenosu – např. baudrate, stop bity, datové bity, parita a samozřejmě i označení portu, na kterém se vytvoří spojení. Stejně tak lze velice rychle ve zdrojovém kódu vytvořit základní kód událostí komponenty, které jsou celkem tři – DataReceived, ErrorRecieved a PinChanged. V testovacím programu byla použita jen první jmenovaná událost.

Zkušební program byl vytvořen jako projekt Windows Forms, tedy ve vizuálním návrhu aplikace. Obsahuje pole pro zadání portu určeného ke komunikaci, několik tlačítek s pevně danými odesílanými hodnotami a TextBox, do kterého se vypisují přijatá data (viz Obr. 8).

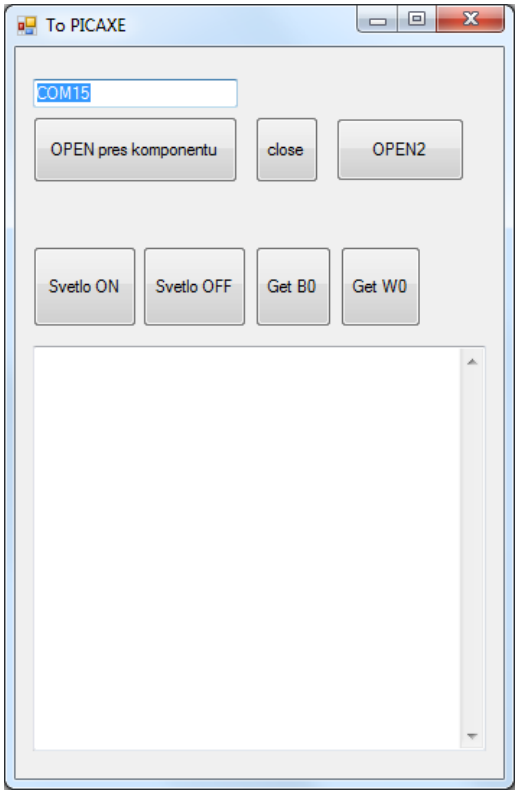
Zaslání dat přes komponentu SerialPort se provede kliknutím na některé z tlačítek v aplikaci. V následující ukázce je vidět, jak dochází k odeslání pokynu rozsvícení druhých světel po kliknutí na příslušné tlačítko:

```
private: System::Void button2_Click(System::Object^
sender, System::EventArgs^  e) {
            d[4]=209; d[5]=7;d[6]=0;d[7]=0;
            serialPort1->Write(d, 0, 12);

            }
```

První čtyři a poslední čtyři hodnoty 12-ti rozměrného pole (podle používaného protokolu) jsou naplněny globálně v předcházející části kódu. Zde se tedy doplní číselné hodnoty do zbývajících (datových) a následně je celé pole dat odesláno prostřednictvím funkce `Write` komponenty `serialPort1` na příslušný sériový port.

Příjem dat je pak řešen v události `DataReceived`, která je automaticky vyvolána, pokud jsou přijata jakákoli data. Tato data jsou pomocí funkce `ReadByte` komponenty `serialPort1` načtena po jednotlivých bytech do pole a následně vypsána do připraveného TextBoxu. Tím uživatel získá přehled o odpovědi mikrokontroléru a může tak snadno zjistit, zda komunikace proběhla úspěšně nebo se vyskytla chyba. Při testování byla očekávána předem známá data, a pokud byla přijata špatná data, dalo se velice rychle chybu odstranit a opětovně otestovat správnou funkčnost celé komunikace.



Obr. 8 - Testovací software využívající komponentu SerialPort

5.2 Windows API

Windows API je API vyvinuté firmou Microsoft pro operační systém Microsoft Windows. Obsahuje procedury, protokoly, funkce, třídy a definice souvisejících datových typů, které lze při vytváření softwaru využívat. Jak jsou funkce ze zdrojového kódu volány určuje právě API. Na použitém programovacím jazyce nezáleží. V Microsoft Windows musí programy komunikovat prostřednictvím Windows API.

Největší výhodou programování za využití Windows API je to, že už od prvního uvedení ve Windows 1.0 zůstává toto API zcela konzistentní a jediné změny jsou v tom, že je celý balík postupně rozšiřován. Například první verze operačního systému Windows podporovala necelých 450 funkcí, v současné době jich jsou již tisíce. Tím je zajištěno, že programátor, zvládající programování pro starší verzi Windows, bude schopen bez problémů pokračovat ve své práci i na nových vydáních tohoto operačního systému od Microsoftu.

Zřejmě nejzásadnější změna ve Windows API nastala v roce 1995 s příchodem Windows 95, ve kterých již byla oproti starším verzím tohoto operačního systému použita 32-bitová architektura (do té doby byla 16-bitová). Došlo například k rozšíření datového typu integer na 32 bitů, nebo některých funkcí. (5 str. 9)

API pro 16-ti bitové verze Windows se označuje jako Win16, analogicky pak existuje Win32 a Win32 pro 64-bitové verze (označované také jako Win64). Všechny 32-bitové verze Windows (nejstarší Windows 95 až nejnovější Windows 8) obsahují tzv. Win16 subsystém, což znamená, že lze na nich spustit aplikace psané pro 16bit. Oproti tomu 64-bitové verze podporují spouštění jen 32 a 64bit aplikací, pro 16-ti bitové se musí používat emulátor.

5.3 Asynchronní komunikace pomocí C++ a Windows API

Hlavním cílem této bakalářské práce bylo vytvořit asynchronní komunikaci po sériové lince. Asynchronní komunikace se od sériové liší tím, že neprobíhá v reálném čase. Po zaslání zprávy na příslušné zařízení může systém dále vykonávat jinou činnost bez čekání na odpověď. Může tak například i zaslat další

instrukce ke zpracování. Ve chvíli, kdy jsou přijata příchozí data, mohou být zařazena do vytvořené fronty, odkud jsou vyzvednuta podle potřeby, nebo mohou vyvolat událost, ve které proběhne ihned jejich zpracování a na tomto základě určená činnost. To je většinou nejen u mobilních robotů žádoucí, jelikož ve spoustě případů není požadavek na vykonávání pouze jedné instrukce, ale více současně.

Pro asynchronní komunikaci byla vytvořena zcela nová třída CComPort, jejíž funkce lze volat. Řešení formou třídy bylo zvoleno proto, aby byl tento napsaný kód použitelný i v dalších projektech, kde se využívá komunikace po sériové lince. Celý zdrojový kód byl tedy napsán pro univerzální použití, přičemž hlavní důraz se kladl na to, aby vše bylo funkční především v průzkumné miniponorce.

Komunikaci přes sériovou linku lze řešit vícero způsoby, nejpoužívanější je ale zřejmě přes tyto API funkce:

- CreateFile – otevření portu a zahájení spojení
- WriteFile – zápis dat na otevřený port
- ReadFile – čtení dat z otevřeného portu

Funkce WriteFile a ReadFile jsou však určeny pro synchronní komunikaci, pro asynchronní přenos je proto potřeba využít jejich odvozeniny – WriteFileEx a ReadFileEx. Tyto funkce disponují speciálním parametrem typu LPOVERLAPPED_COMPLETION_ROUTINE, který odkazuje na dokončovací rutinu, pokud je zápis, resp. čtení dokončeno nebo přerušeno. Tato rutina je podrobněji popsána v kapitole 5.3.3.

5.3.1 Navázání spojení a nastavení portu

Prvotní fází komunikace je otevření požadovaného portu. V třídě CComPort byla vytvořena funkce OpenPort s 5 parametry – viz kód:

```
long CComPort::OpenPort(char* portName, DWORD dwBaudRate,
byte byByteSize, byte byParity, byte byStopBits)
```

První parametr určuje jméno portu, na který je potřeba se připojit. Toto připojení je provedeno přes API funkci CreateFileA:

```
serial_port = CreateFileA(portName, GENERIC_READ | GENERIC_WRITE
| SYNCHRONIZE, 0, 0, OPEN_EXISTING, FILE_FLAG_OVERLAPPED , 0);
```

Port musí být otevřen pro čtení (`GENERIC_READ`) i zápis (`GENERIC_WRITE`) a také nastaven do asynchronního režimu (`FILE_FLAG_OVERLAPPED`). Funkce CreateFileA má návratovou hodnotu typu HANDLE, v tomto případě je objekt handle `serial_port`. Tento objekt je deklarován jako globální, aby ho bylo možné využít i ve více funkcích – například v druhé fázi čtení z portu je tato vlastnost využita.

Po kontrole, zda je port otevřen, je potřeba správně nastavit celou komunikaci po sériové lince. Nejprve dojde k zjištění aktuálního nastavení pomocí API funkce `GetCommState`, která načtené hodnoty ukládá do instance struktury DCB pojmenované jako `dcb`. Struktura DCB obsahuje 21 členů, zde však postačilo nastavit čtyři z nich:

```
dcb.BaudRate    = dwBaudRate;
dcb.ByteSize    = byByteSize;
dcb.Parity      = byParity;
dcb.StopBits    = byStopBits;
```

Hodnoty přiřazované k členům jsou předávány v parametrech funkce OpenPort. Dokončení nastavení portu se provede API funkcí `SetCommState`:

```
SetCommState(serial_port,&dcb)
```

Pokud vše proběhlo úspěšně, tedy připojení se k portu, získání původního nastavení a nové nastavení, funkce vrátí hodnotu nula.

5.3.2 Zápis dat

K zápisu dat na sériový port je vytvořena funkce `Write(unsigned char* buff, int nBuff)`, ve které je použita API funkce WriteFileEx. Tato funkce má celkem pět parametrů, z toho dva nepovinné. Posledním parametrem, a také druhým nepovinným, je lpCompletionRoutine. Ten se využívá v případě, kdy je potřeba přejít po dokončení nebo přerušení čtení do jiné funkce. Stejný parametr má i API funkce ReadFileEx používaná pro čtení. U zápisu dat však tento parametr nebyl použit. Vykonání zápisu dat na port provede následující kód:

```
WriteFileEx(serial_port, buff, nBuff, &overlapped, NULL);
```

První parametr je handle na otevřený sériový port, následují předávaná data, délka předávaných dat zjištěná pomocí `sizeof(buff)`, ukazatel na strukturu overlapped a jak již bylo zmíněno, poslední parametr nebyl využit a je tedy nastaven na hodnotu NULL.

5.3.3 Čtení dat

Pro čtení dat je ve třídě CComPort určena funkce `Read(unsigned char* buff, int nBuff)`, využívající API funkci ReadFileEx. Ta má stejně jako WriteFileEx pět parametrů, tentokrát je však využit i poslední parametr. Zápis funkce ReadFileEx vypadá následovně:

```
ReadFileEx(serial_port, buff, 1000, &overlapped, readComplete)
```

Předně je potřeba funkci předat handle, tedy port, ze kterého se data mají číst. Dalším parametr je buffer, do kterého se přijatá data uloží, v tomto případě se jedná o pole typu char délky 1000 bytů. Počet načítaných bytů určuje třetí parametr funkce. Následuje ukazatel na strukturu overlapped, která obsahuje informace využívané při asynchronní komunikaci. A poslední parametr odkazuje na tzv. callback funkci, která se vykoná po dokončení čtení. Jelikož je celá komunikace v asynchronním režimu, čtení dat může probíhat kdykoli. Proto se po prvním čtení přejde ihned k dalšímu čtení, které následně odkazuje samo na sebe a provádí se tedy nepřetržitě až do ukončení spojení. Tím je zajištěno, že přijatá data budou ihned zpracována a připravena k dalšímu použití.

Callback funkce `readComplete(const DWORD errorCode, const DWORD bytesCopied, OVERLAPPED* overlapped)` opět využívá API funkci ReadFileEx, a to ve stejné podobě, jako funkce Read.

První parametr funkce readComplete vrací číslo chyby, která při přenosu nastala. Pokud k žádné chybě nedošlo, errorCode bude obsahovat nulovou hodnotu. Při testování softwaru se však objevoval chybový kód 995, který označuje, že při komunikaci došlo pravděpodobně k přerušení spojení. To bylo zapříčiněno tím, že mikrokontrolér PICAXE nebyl schopný odesílat data přes

sériovou linku tak rychle, jak byla na straně nadřazeného počítače čtena a čtení bylo proto nutné zpozdit o cca 30 milisekund. Do budoucna by však bylo vhodné tento problém řešit jiným, vhodnějším způsobem. Druhý parametr podává informaci o tom, kolik bytů bylo celkem přeneseno, což můžeme využít pro ověření, zda opravdu byla načtena všechna data, která jsou pro další činnost softwaru potřeba.

5.3.4 Ukončení spojení

Na konci celé komunikace je potřeba ukončit navázané spojení a otevřený port uzavřít. Až po uzavření portu mohou jiné procesy tento port otevřít pro své potřeby (přistupovat k němu), do té doby je blokován pro existující spojení. K tomuto účelu je součástí třídy CComPort funkce ClosePort. Ta pomocí API funkce CloseHandle ukončí spojení – viz ukázka:

```
CloseHandle(serial_port);
```

Tím je komunikace ukončena, a pokud je opět potřeba navázat spojení, musí se znovu zavolat funkce OpenPort.

6 Praktické otestování softwaru

K praktickému otestování softwaru přímo v miniponorce při reálném ponoru bohužel před odevzdáním této bakalářské práce nedošlo. Při oživování plavidla před závěrečným testem došlo k závadě na jednom z motorů a nebylo v časových možnostech zajistit rychlou opravu. Z tohoto důvodu se přistoupilo k otestování na testovací desce s mikrokontrolérem PICAXE 18M2 (viz Obr. 3) a připojeném stolním počítači.

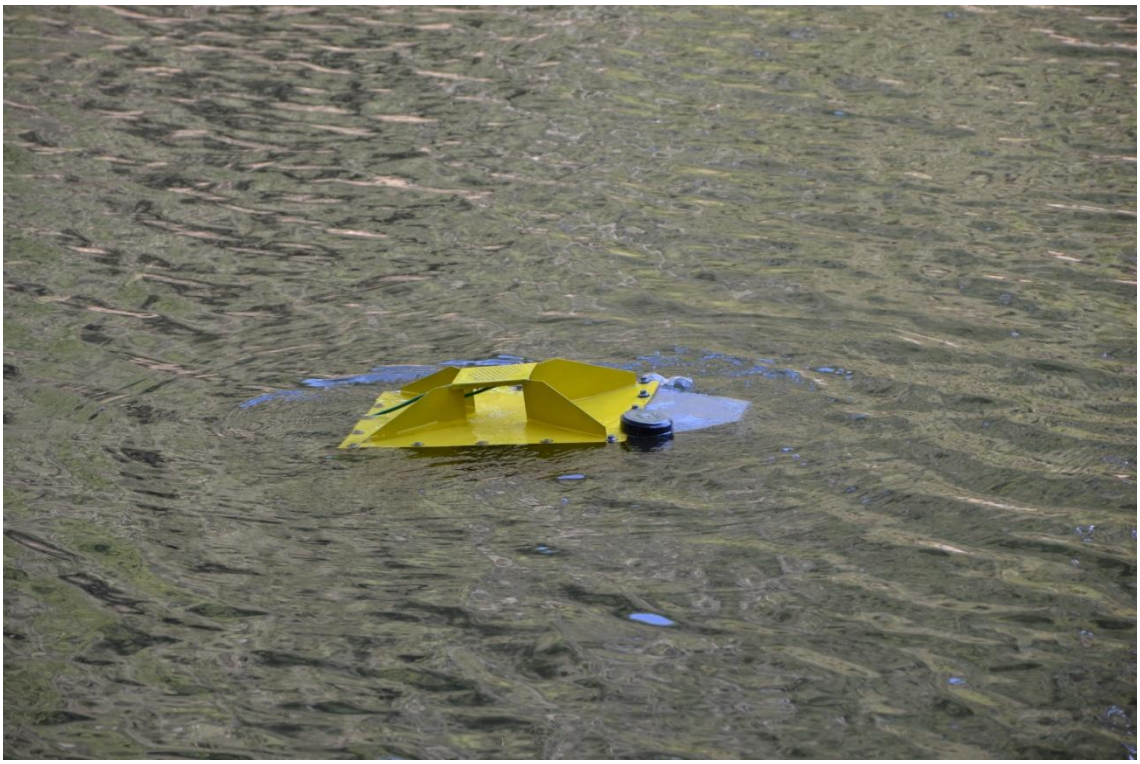
Připojení, zápis i čtení přes sériovou linku probíhalo bez komplikací, jediným problémem je pomalejší odezva mikroprocesoru, kterou bude do budoucna vhodné vyřešit. To povede zřejmě na úpravu řídícího programu pro PICAXE nebo jiným čekáním na odpověď v nadřazeném počítači.

Software mikrokontroléru zpracovává data bez potíží, v průběhu testování nebyla zaznamenána žádná chyba.

7 Další možnosti rozvoje projektu miniponorky

Projekt miniponorky je rozhodně projektem, který se dá do budoucna dále rozvíjet. V plánu je několik dalších zařízení, které učiní plavidlo více přínosným – GPS určování polohy, teplotní čidla pro měření teploty uvnitř ponorky i vody, nebo jednoduchý sonar.

Je však důležité zvážit i výměnu součástí, které jsou již v současnosti namontovány. Jedná se především o úpravu rour kolem motorů, aby lépe odváděly vodu a motory mohly vyvinout vyšší rychlost plavidla, ale také o výměnu samotných motorů, z nichž jeden po takřka ročním nevyužití přestal fungovat. Za zvážení stojí i změna lopatek motorů, zda by neměly vliv na účinnost motorů. Dalším prvkem pro výměnu je osvětlení, ve kterém došlo v průběhu minulých ponorů ke zkratům a tím k jeho vyřazení z činnosti.



Obr. 9 - ponorka při jednom z ponorů v akademickém roce 2011/2012

8 Závěr

Pomocí této bakalářské práce se podařilo seznámit s problematikou komunikace přes sériový port zejména v prostředí Windows a s možnostmi realizace tohoto spojení. Důležité bylo zejména pochopení Windows API a práce s ním.

Byl navržen a použit univerzální komunikační protokol, který má uplatnění nejen v projektu průzkumné miniponorky, ale může jej mít například i u jiných robotických zařízení využívajících přenos dat přes sériový port.

Hlavním přínosem této práce je realizace zcela nové komunikace ze strany nadřazeného počítače pomocí funkcí Windows API, tedy nízkoúrovňového softwaru, který by měl být bez problémů použitelný i v budoucnosti na nových verzích operačního systému Windows díky tomu, že Windows API zůstává poměrně konzistentní a je stále rozšiřováno při zachování již stávajících funkcí.

Byl také vytvořen nový řídící software pro mikrokontrolér PICAXE 18M2, který je schopný pracovat pomocí navrženého komunikačního protokolu a v režimu „příkaz – odpověď".

Bibliografie

1. **Novák, Petr.** *Mobilní roboty - pohony, senzory, řízení.* Praha : BEN - technická literatura, 2005. str. 248. ISBN 80-7300-141-1.

2. **Dlouhý, Martin a Winkler, Zbyněk.** Co je to robot? *Robotika.cz.* [Online] 22. 1 2005. [Citace: 1. 5 2013.] http://robotika.cz/guide/robot/cs.

3. **Matoušek, David.** *Udělejte si z PC - generátor, čítač, převodník, programátor...* Praha : BEN - technická literatura, 2001. str. 175. ISBN 80-7300-036-9.

4. **Pužmanová, Rita.** *TCP/IP v kostce.* České Budějovice : Kopp nakladatelství, 2009. str. 619. ISBN 978-80-7232-388-3.

5. **Petzold, Charles.** *Programování ve Windows.* Praha : Computer Press, 1999. str. 1216. ISBN 80-7226-206-8.

Příloha A – tabulka příkazových hodnot

Hodnota	Příkaz
2001	pohyb vpřed
2002	pohyb vzad
2003	pohyb vlevo
2004	pohyb pravo
2005	stop
2006	světlo 1 vypnout
2007	světlo 1 zapnout
2008	světlo 2 vypnout
2009	světlo 2 zapnout
2010	čerpadlo – zapnutí napouštění
2011	čerpadlo – vypnutí napouštění
2012	čerpadlo – zapnutí vypouštění
2013	čerpadlo – vypnutí vypouštění
2014	dotaz na stav baterií
2015	dotaz na teplotu uvnitř ponorky
2016	dotaz na teplotu vody
2017	dotaz na hloubku